

---

# COMP5338 A2

---

Albert Huang 530215563

## 1 Introduction

Climate issue has been a major problem for all regions. In this assignment, three *json* files are given with respective information of climates, coordination and CO2 emissions of some cities or of some regions. The goal of this assignment is to test the ability of modeling MongoDB data and writing some complex aggregation workloads.

## 2 Schema Design

In my design, I have three collections, *climates*, *coords* and *co2*. All of them are in a database called *emissions*.

### 2.1 climates

This collection are directly derived from *climates.json*. Each document has a "\_id", a "city" field showing the city name, a "region" field showing the region the city belongs to, a "monthlyAvg" field cotain climates data of 12 months. Each month has "high", "low", "dryDays", "snowDays", and "rainfall" shwoing data of different climates aspects.

```
1 {
2   "_id": 9,
3   "city": "Berlin",
4   "region": "Germany",
5   "monthlyAvg": [
6     {
7       "high": 3,
8       "low": -1,
9       "dryDays": 7,
10      "snowDays": 10,
11      "rainfall": 42
12    },
13    {
14      "high": 3,
15      "low": -2,
16      "dryDays": 10,
17      "snowDays": 10,
18      "rainfall": 36
19    },
20    {
21      "high": 8,
22      "low": 1,
23      "dryDays": 11,
24      "snowDays": 5,
25      "rainfall": 30
26    },
27    {
28      "high": 13,
29      "low": 4,
30      "dryDays": 11,
31      "snowDays": 2,
32      "rainfall": 36
33    },
34    {
35      "high": 19,
36      "low": 9,
37      "dryDays": 13,
38      "snowDays": 0,
39      "rainfall": 48
40    },
41    {
42      "high": 21,
43      "low": 12,
44      "dryDays": 9,
45      "snowDays": 0,
46      "rainfall": 72
47    },
48  ]
49 }
```

Figure 1: Example Document part 1

```

48     {
49         "high": 24,
50         "low": 14,
51         "dryDays": 14,
52         "snowDays": 0,
53         "rainfall": 36
54     },
55     {
56         "high": 24,
57         "low": 14,
58         "dryDays": 15,
59         "snowDays": 0,
60         "rainfall": 45
61     },
62     {
63         "high": 19,
64         "low": 11,
65         "dryDays": 12,
66         "snowDays": 0,
67         "rainfall": 39
68     },
69     {
70         "high": 14,
71         "low": 7,
72         "dryDays": 13,
73         "snowDays": 0,
74         "rainfall": 24
75     },
76     {
77         "high": 7,
78         "low": 3,
79         "dryDays": 9,
80         "snowDays": 4,
81         "rainfall": 51
82     },
83     {
84         "high": 4,
85         "low": 0,
86         "dryDays": 8,
87         "snowDays": 9,
88         "rainfall": 51
89     }
90 ]
91 }

```

Figure 2: Example Document part 2

## 2.2 coords

This collection also directly derived from its original dataset *coords.json*. Each document has a "id" field, a "city\_id" that link this document to a document in *climates* collection with the same id, a "city" showing the city's name. Also, a "latitude", a "longitude", and a "elevation" which represent the detail data of corresponding coordinates.

```
1  {
2    "_id": 17,
3    "city_id": 16,
4    "city": "Budapest",
5    "latitude": 47.5,
6    "longitude": 19.04,
7    "elevation": 102
8  }
```

Figure 3: Example Document

### 2.3 co2

This collection uses a pre-processed dataset called *processed\_co2.json* which is a reconstructed version of *co2.json* that separate each region and its embedded data into different documents. Each document represent a region's CO2 emission data. It has a "\_id", a "region" showing the region name, a "iso\_code" showing the iso code of that region, and a "data" which is a list embeds with emission related data of each year. Due to the length of a document, below only showing parts of the document for reference.

```

1 {
2   "_id": 1,
3   "region": "Argentina",
4   "iso_code": "ARG",
5   "data": [
6     {
7       "year": 1950,
8       "population": 17017748,
9       "gdp": 136328019968,
10      "cement_co2": 0.777737021446228,
11      "cement_co2_per_capita": 0.0457015223801136,
12      "co2": 29.921192169189453,
13      "co2_growth_abs": 14.55656623840332,
14      "co2_growth_prct": 94.74076843261719,
15      "co2_including_luc": 42.12978744506836,
16      "co2_including_luc_growth_abs": 34.61971664428711,
17      "co2_including_luc_growth_prct": 460.9770812988281,
18      "co2_including_luc_per_capita": 2.4756381511688232,
19      "co2_including_luc_per_gdp": 0.30903249979019165,
20      "co2_per_capita": 1.7582345008850098,
21      "co2_per_gdp": 0.21947939693927765,
22      "coal_co2": 3.561408042907715,
23      "coal_co2_per_capita": 0.20927610993385315,
24      "cumulative_cement_co2": 10.510852813720703,
25      "cumulative_co2": 555.9810180664062,
26      "cumulative_co2_including_luc": 7059.30078125,
27      "cumulative_coal_co2": 309.18902587890625,
28      "cumulative_flaring_co2": 0,
29      "cumulative_gas_co2": 20.849384307861328,
30      "cumulative_luc_co2": 7104.1455078125,
31      "cumulative_oil_co2": 215.4317626953125,
32      "flaring_co2": 0,
33      "flaring_co2_per_capita": 0,
34      "gas_co2": 0,
35      "gas_co2_per_capita": 0,
36      "land_use_change_co2": 12.208595275878906,
37      "land_use_change_co2_per_capita": 0.7174036502838135,
38      "oil_co2": 25.582048416137695,
39      "oil_co2_per_capita": 1.503256916999817,
40      "share_global_cement_co2": 1.1625432968139648,
41      "share_global_co2": 0.5046292543411255,
42      "share_global_co2_including_luc": 0.3511865437030792,
43      "share_global_coal_co2": 0.09237605333328247,
44      "share_global_cumulative_cement_co2": 1.1694997549057007,
45      "share_global_cumulative_co2": 0.24092479050159454,
46      "share_global_cumulative_co2_including_luc": 1.1034259796142578,
47      "share_global_cumulative_coal_co2": 0.15664011240005493,

```

Figure 4: Example Document part 1

```

48     "share_global_cumulative_flaring_co2": 0,
49     "share_global_cumulative_gas_co2": 0.4289860725402832,
50     "share_global_cumulative_luc_co2": 1.7177814245224,
51     "share_global_cumulative_oil_co2": 0.7857019305229187,
52     "share_global_flaring_co2": 0,
53     "share_global_gas_co2": 0,
54     "share_global_luc_co2": 0.20122717320919037,
55     "share_global_oil_co2": 1.6240003108978271,
56     "share_of_temperature_change_from_ghg": 1.957290530204773,
57     "temperature_change_from_ch4": 0.0025573191232979298,
58     "temperature_change_from_co2": 0.005146034527570009,
59     "temperature_change_from_ghg": 0.007861865684390068,
60     "temperature_change_from_n2o": 0.0001585122269702703
61 },
62 {
63     "year": 1951,
64     "population": 17354614,
65     "gdp": 141645234176,
66     "cement_co2": 0.77046799659729,
67     "cement_co2_per_capita": 0.0443955697119236,
68     "co2": 34.96291732788086,
69     "co2_growth_abs": 5.04172420501709,
70     "co2_growth_prct": 16.850006103515625,
71     "co2_including_luc": 57.51186752319336,
72     "co2_including_luc_growth_abs": 15.382080078125,
73     "co2_including_luc_growth_prct": 36.51116943359375,
74     "co2_including_luc_per_capita": 3.3139238357543945,
75     "co2_including_luc_per_gdp": 0.40602755546569824,
76     "co2_per_capita": 2.014617919921875,
77     "co2_per_gdp": 0.246834397315979,
78     "coal_co2": 5.4703521728515625,
79     "coal_co2_per_capita": 0.315210223197937,
80     "cumulative_cement_co2": 11.281320571899414,
81     "cumulative_co2": 590.9439086914062,
82     "cumulative_co2_including_luc": 7116.81298828125,
83     "cumulative_coal_co2": 314.65936279296875,
84     "cumulative_flaring_co2": 0,
85     "cumulative_gas_co2": 22.267351150512695,
86     "cumulative_luc_co2": 7126.6943359375,
87     "cumulative_oil_co2": 242.7358856201172,
88     "flaring_co2": 0,
89     "flaring_co2_per_capita": 0,
90     "gas_co2": 1.4179680347442627,
91     "gas_co2_per_capita": 0.08170553296804428,
92     "land_use_change_co2": 22.548952102661133,

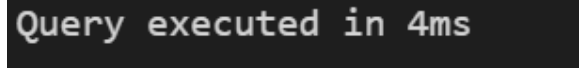
```

Figure 5: Example Document part 2

### 3 Index and Query Execution

#### 3.1 W1

The original runtime is 4ms.



```
Query executed in 4ms
```

Figure 6: Run Time

Since the workload only require the data in year 2022, a **index of year** could be created so when running, it doesn't have to visit all embedded documents, only visit those year is 2022 through looking up in index. In this way, performance increases four times.



```
Query executed in 1ms
```

Figure 7: Run Time with Index

### 3.2 W2

The original runtime of the query is 3ms, and the execution plan is shown below.



```
Runtime: 3 ms
```

Figure 8: Run Time of W2

```

{
  explainVersion: '1',
  stages: [
    {
      '$cursor': [Object],
      nReturned: Long('0'),
      executionTimeMillisEstimate: Long('0')
    },
    {
      '$lookup': [Object],
      totalDocsExamined: Long('0'),
      totalKeysExamined: Long('0'),
      collectionScans: Long('0'),
      indexesUsed: [],
      nReturned: Long('0'),
      executionTimeMillisEstimate: Long('0')
    },
    {
      '$match': [Object],
      nReturned: Long('0'),
      executionTimeMillisEstimate: Long('0')
    },
    {
      '$sort': [Object],
      totalDataSizeSortedBytesEstimate: Long('0'),
      usedDisk: false,
      spills: Long('0'),
      spilledDataStorageSize: Long('0'),
      nReturned: Long('0'),
      executionTimeMillisEstimate: Long('0')
    },
    {
      '$project': [Object],
      nReturned: Long('0'),
      executionTimeMillisEstimate: Long('0')
    }
  ],

```

Figure 9: Execution Plan of W2



Since W2 requires join based on city, it would be beneficial to have **indexes of city** on both collections. The upgraded runtime performance can reach 1ms.



Runtime: 1 ms

Figure 10: Run Time with indexes W2

### 3.3 W3

With the existed index "**city\_1**" in *coords* collection and *climates* collection, and the existed index "**data.year\_1**" in *co2* collection, the performance of W3 query is already optimised at a certain level with a approximately 1ms runtime. This is because the query of W3 is heavily involves with data.year, city names which related to above indexes.

To further improve the performance, it is possible to add a **index of region** in *climates* collection to improve the performance of joins using region.

### 3.4 W4

The original runtime of W4 is 26ms.



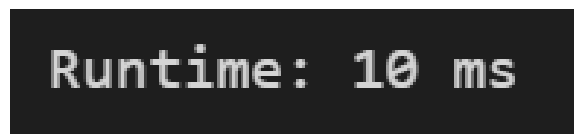
Runtime: 26 ms

Figure 11: Runtime W4

No index is created for this workload. Since *\$unwind* is used, no *\$match* is used, and the process is basic calculation on fields then create new collection, the need for index is not necessary.

### 3.5 W5

The original runtime of W5 is 10ms.



Runtime: 10 ms

Figure 12: Original Runtime of W5

Since the Workload requires a *\$match* command of year 2022, it would be beneficial to create a **index of year** in this collection. After the creation of the index, the performance of workload increase and index is utilized as shown in the execution plan.

Runtime: 4 ms

Figure 13: Runtime of W5 with Index

```
"indexFilterSet": false,
"parsedQuery": {
  "year": {
    "$eq": 2022
  }
},
"queryHash": "8C460F27",
"planCacheKey": "AAE4C0A5",
"maxIndexedOrSolutionsReached": false,
"maxIndexedAndSolutionsReached": false,
"maxScansToExplodeReached": false,
"winningPlan": {
  "stage": "FETCH",
  "inputStage": {
    "stage": "IXSCAN",
    "keyPattern": {
      "year": 1
    },
    "indexName": "year_1",
    "isMultiKey": false,
    "multiKeyPaths": {
      "year": []
    },
    "isUnique": false,
    "isSparse": false,
    "isPartial": false,
    "indexVersion": 2,
    "direction": "forward",
    "indexBounds": {
      "year": [
        "[2022, 2022]"
      ]
    }
  }
},
"rejectedPlans": []
},
```

Figure 14: Execution Plan with Index

### 3.6 W6

The original runtime of W6 is 9ms.



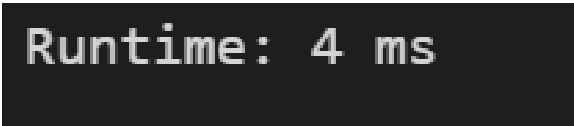
Runtime: 9 ms

Figure 15: Original Runtime of W6

There is *\$match* for elevation in the *coords* collection, therefore a possible performance improvement is to create a **index of elevation**. In this way, the workload is using two indexes: "elevation\_1" and "year\_1" of *analytics1* collection. The execution plan below shows the utilisation of indexes. And the runtime is reduced significantly.

```
"inputStage": {
  "stage": "IXSCAN",
  "nReturned": 0,
  "executionTimeMillisEstimate": 0,
  "works": 1,
  "advanced": 0,
  "needTime": 0,
  "needYield": 0,
  "saveState": 1,
  "restoreState": 1,
  "isEOF": 1,
  "keyPattern": {
    "elevation": 1
  },
  "indexName": "elevation_1",
  "isMultiKey": false,
  "multiKeyPaths": {
    "elevation": []
  },
  "isUnique": false,
  "isSparse": false,
  "isPartial": false,
  "indexVersion": 2,
  "direction": "forward",
  "indexBounds": {
    "elevation": [
```

Figure 16: Execution Plan with Indexes




Runtime: 4 ms

Figure 17: Runtime of W6 with Indexes

### 3.7 W7

The runtime of W7 is 32ms.



```
Runtime: 32 ms
```

Figure 18: Enter Caption

Similar to W4, the workload 7 is simple match of a range of year and calculation, with index of year implemented before, no other index is needed.

### 3.8 W8

The runtime of W8 is 43ms.



```
Runtime: 43 ms
```

Figure 19: Enter Caption

The solution use *\$unwind* in the beginning on *co2* collection, and the later process of this workload uses only data after the *\$unwind* command, therefore no index on the collection is utilized.

## 4 Conclusion

The assignment showcases the knowledge of modeling MongoDB data, designing complex aggregation workloads, creating index and reviewing performance of query. And through practises, it can be confident to say that the skills related to MongoDB is greatly enhanced. For the final result, all workloads are completed as required.